Assignment No: 04

Name of student:

Roll No:

Practical Batch:

Title of the Assignment: Constraint Satisfaction Problem

Problem statement: Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.

Objective:

- To study Constraint Satisfaction Problem using Branch and Bound and Backtracking.
- To implement graph coloring problem using CSP.

Theory:

Constraint Satisfaction Problem:

A constraint satisfaction problem (CSP) is defined by a set of variables, x1, x2,….xn , and a set of constraints, c1,c2….cm. Each variable has a nonempty domain of possible values. Each constraint involves some subset of the variables and specifies the allowable combinations of values for that subset. A state of the problem is defined by an assignment of values to some or all of the variables, {xi=vi,xj=vj,…}. An assignment that does not violate any constraints is called a consistent or legal assignment. A complete assignment is one in which every variable is mentioned, and a solution to a CSP is a complete assignment that satisfies all the constraints.

In constraint satisfaction, domains are the spaces where the variables reside, following the problem specific constraints. These are the three main elements of a constraint satisfaction technique. The constraint value consists of a pair of **{scope, rel}**. The **scope** is a tuple of variables which participate in the constraint and **rel** is a relation which includes a list of values which the variables can take to satisfy the constraints of the problem.

Graph Coloring: The problem where the constraint is that no adjacent sides can have the same color.

Graph coloring is the procedure of assignment of colors to each vertex of a graph G such that no adjacent vertices get same color. The objective is to minimize the number of colors while coloring a graph. The smallest number of colors required to color a graph G is called its chromatic number of that graph. Graph coloring problem is a NP Complete problem.

Method to Color a Graph:

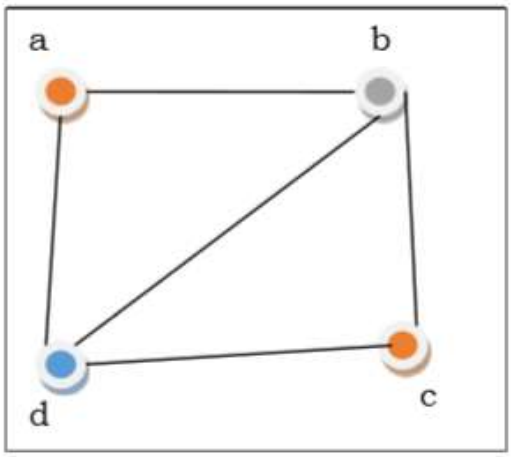
The steps required to color a graph G with n number of vertices are as follows −

Step 1 − Arrange the vertices of the graph in some order.

Step 2 − Choose the first vertex and color it with the first color.

Step 3 − Choose the next vertex and color it with the lowest numbered color that has not been colored on any vertices adjacent to it. If all the adjacent vertices are colored with this color, assign a new color to it. Repeat this step until all the vertices are colored.

Example:



In the above figure, at first vertex ***a*** is colored red. As the adjacent vertices of vertex ***a*** are again adjacent, vertex ***b*** and vertex ***d*** are colored with different color, gray and blue respectively. Then vertex ***c*** is colored as red as no adjacent vertex of ***c*** is colored red. We could color the graph by 3 colors. Hence, the chromatic number of the graph is 3.

Pseudocode:

```
def isSafeToColor(graph, color):
    for i in range(V):
        for j in range(i + 1, V):
            if graph[i][j] == 1 and color[j] == color[i]:
                return False
    return True
def printColorArray(color):
    print("Solution colors are: ")
    for i in range(len(color)):
        print(color[i], end=" ")
```

```
def graphColoring(graph, m, i, color):
    if i == V:
        if isSafeToColor(graph, color):
            printColorArray(color)
            return True
        return False
    for j in range(1, m + 1):
        color[i] = j
        if graphColoring(graph, m, i + 1, color):
            return True
        color[i] = 0
return false
```

Pseudocode: Backtracking search

```
function Backtracking-Search(csp) returns solution/failure
    return Recursive-Backtracking({ }, csp)
function Recursive-Backtracking(assignment, csp) returns soln/failure
    if assignment is complete then return assignment
   var←Select-Unassigned-Variable(Variables[csp], assignment, csp)
for each value in Order-Domain-Values(var, assignment, csp) do
        if value is consistent with assignment given Constraints[csp] then
            add {var = value} to assignment
            result ←Recursive-Backtracking(assignment, csp)
            if result ≠ failure then return result
            remove {var = value} from assignment
return failure
```

Backtracking = DFS + variable-ordering + fail-on-violation
Backtracking enables us the ability to solve a problem as big as 25-queens

Conclusion:

Constraint satisfaction problems (CSP)s are mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations. CSPs represent the entities in a problem as a homogeneous collection of finite constraints over variables, which is solved by constraint satisfaction methods. CSPs are the subject of intense research in both artificial intelligence and operations research, since the regularity in their formulation provides a common basis to analyze and solve problems of many unrelated families. CSPs often exhibit high complexity, requiring a combination of heuristics and combinatorial search methods to be solved in a reasonable time.
We have implemented graph coloring problem using CSP.

Oral questions:

1. Consider a problem of preparing a schedule for a class of student. What type of problem is this?
2. Define Constraint satisfaction problem.
3. What are the variations on CSP formalism?
4. What are the issues that need to be addressed for solving CSP efficiently?
5. Explain Backtracking search in CSP for n queens.
6. Detail the concepts of backtracking and constrain propagation and solve the n-queen problem using these algorithms.